REMARKS

The Examiner's remarks in response to the Applicant's arguments dated 12/8/2008 focus on the Examiner's overly expansive definition of the term "High Level Programming Language" as used in claims 1-17. The Examiner's statements that "Applicant's specification does not provide any clarification of what constitutes a 'high level' program/language" show a further misunderstanding as to the claimed scope of the application. Claims 1 and 8 both explicitly claim a "programmable device capable of being programmed using a high level programming language and capable of being reprogrammed." The Examiner appears to be equating "high level program/language" (see Page 12 line 18 of the February 11th office action) with the claimed "programmed using a high level programming language" (see claims 1 and 8 of the present application.)

The Examiner's comments further show a misunderstanding of the term "high level programming language" as would be understood by one having ordinary skill in the art. The Examiner has argued that the term "high level programming language" is overly broad. The Applicant maintains that the term "high level programming language" has a known definition which would be understood by a person having ordinary skill in the art. As further evidence that the term "high level programming language" has a known definition, the Applicant has provided the below source having a definition of a high level programming language.

New Perspectives Computer Concepts, © 2009, Course Technology, Cengage Learning (ISBN-10: 1-4239-2518-1) defines a high level programming language as a programming language which "uses command words and grammar based on human languages to provide what computer scientists call a level of abstraction that hides the underlying low-level assembly or machine language. High-level languages, such as COBOL, BASIC, Java, and C, make the programming process easier by replacing unintelligible strings of 1s and 0s or cryptic assembly commands with understandable commands, such as PRINT and WRITE. High-Level language commands eliminate many lines of code by substituting a single hegh0level command for multiple low-level commands."

As evidenced by the above definition, "high level programming language" has a known meaning which would be understood by a person having ordinary skill in the art.

If the Examiner wishes to maintain his assertion that "high level programming language" can have further definitions beyond the commonly known meaning, the Applicant respectfully requests that the Examiner take official notice, and provide justification for his assertion, in a new non-final office action.

Furthermore, the context of the paragraphs 18 and 23 of the applicant's specification, as well as the plain language of the claims, makes it clear that the term "high level programming language" as used here refers to a specific type of programming language, and not to a program operating at a high level as is asserted by the Examiner. (See page 12 of the Office Action dated 2/11/09).

Since the device of Lui does not disclose a programmable device capable of being programmed using a high level programming language, as would be understood by one of ordinary skill in the art, nor does it make such a device obvious, the combination of admitted prior art in view of Lui would also not render such a device obvious.

The Examiner has rejected claims 8, 9, 11, 12, 14, 15, and 17 under 35 USC 103(a) as being unpatentable over admitted prior art (APA) in view of Lui (US 5337413). The Examiner's rejection relies on an erroneously expansive definition of the term "high level programming language." As described above, Lui does not teach or suggest a programmable device capable of being programmed in a high level programming language as the term high level programming language would be understood by a person having ordinary skill in the art. As such, the combination of APA in view of Lui does not teach or suggest the subject matter of claims 8, 9, 11, 12, 14, 15, and 17.

The Examiner has additionally rejected claims 1-6, 13, and 16 under 35 USC 103(a) as being unpatentable over APA in view of Lui, in further view of Alexander (US 6701402). The Examiner's suggested combination suffers from the same defect described above with regard to claims 8, 9, 11, 12, 14, 15, and 17. The additional of Alexander fails to correct this defect, and as such claims 1-6, 13, and 16 are allowable for the reasons described above.

Therefore, claims 1-6, 8, 9, and 11-17 are in condition for allowance, and the applicant respectfully requests that the examiner withdraw his rejections. The Applicant believes that no fees are due at this time, however the Commissioner is authorized to charge Deposit Account No. 08-0385, in the name of Hamilton Sundstrand Corporation

for any additional fees or to credit the account for any overpayment. Therefore, favorable reconsideration and allowance of this application is respectfully requested.

Respectfully Submitted,

CARLSON, GASKEY & OLDS, P.C.

/Theodore W. Olds/

Theodore W. Olds
Registration No. 33,080
400 West Maple, Suite 350
Birmingham, Michigan 48009
Telephone: (248) 988-8360

Dated: April 9, 2009

APPENDIX

Parsons et al., June Jamrich, *"New Perspectives Computer Concepts,"* © 2009, Course Technology, Cengage Learning, Boston, MA (ISBN-10: 1-4239-2518-1), pp. 676.

NEW PERSPECTIVES

COMPUTER CONCEPTS

JUNE JAMRICH PARSONS • DAN OJA

EDITION

COMPREHENSIVE

COURSE TECHNOLOGY
CENGAGE Learning

New Perspectives on Computer Concepts, 11th Edition, Comprehensive

June Jamrich Parsons, Dan Oja

Executive Editor: Marie Lee

Senior Product Manager: Kathy Finnegan

Product Manager: Erik Herman

Associate Product Manager: Brandi Henson

Developmental Editor: Deb Kaufmann

Editorial Assistant: Leigh Robbins

Technology Project Manager: Jim Ruggiero

Director of Marketing: Cheryl Costantini

Marketing Manager: Ryan DeGrote

Marketing Specialist: Jennifer Hankin

Senior Content Project Manager: Jennifer Goguen McGrail

Photo Researcher: Abby Reip

Art Director: Marissa Falco

Cover Designer: Joel Sadagursky

BookOnCD Technician: Keefe Crowley

BookOnCD Development: MediaTechnics Corp.

Prepress Production: GEX Publishing Services

ISBN-13: 978-1-4239-2518-7
ISBN-10: 1-4239-2518-1

Course Technology
25 Thomson Place
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil and Japan. Locate your local office at: international.cengage.com/region

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

For your lifelong learning solutions, visit course.cengage.com

Purchase any of our products at your local college store or at our preferred online store www.ichapters.com

Printed in the United States of America
1 2 3 4 5 6 7 8 9 12 11 10 09 08

PROGRAMMING LANGUAGES AND PARADIGMS

What is a programming language? A **programming language**, or computer language, is a set of keywords and grammar rules designed for creating instructions that a computer can ultimately process or carry out. Most people are familiar with names of popular programming languages, such as BASIC, C, Pascal, FORTRAN, Java, and COBOL. But many other programming languages, such as 8088 assembly, FORTH, LISP, and APL, remain relatively unknown to the general public.

The program you wrote at the beginning of the chapter to change the text color displayed on your computer screen was written in DOS scripting language, which is supported in most versions of Windows.

Just as an English sentence is constructed from various words and punctuation marks that follow a set of grammar rules, each instruction for a computer program consists of keywords and parameters that are held together by a set of rules. A **keyword**, or command, is a word with a predefined meaning for the compiler or interpreter that translates each line of program code into machine language. Keywords for the Pascal computer language include WRITE, READ, IF...THEN, and GOSUB. The Change.bat program you wrote used keywords such as ECHO, SET, IF, PAUSE, and COLOR.

Keywords can be combined with specific **parameters**, which provide more detailed instructions for the computer to carry out. Keywords and parameters are combined with punctuation according to a series of rules called **syntax**, as shown in Figure 12-3.

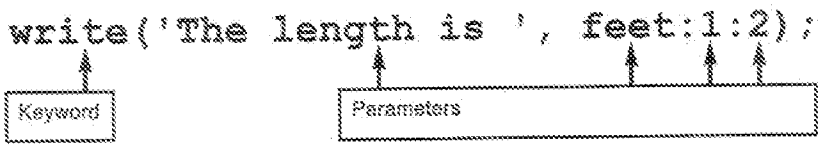

FIGURE 12-3
An instruction for a computer program consists of keywords and parameters, formed into sentence-like statements according to a set of syntax rules.

How are programming languages categorized? Programming languages are categorized in several ways. They can be divided into two major categories: low-level languages and high-level languages. They are also categorized by generation and by paradigm.

What is a low-level language? A **low-level language** typically includes commands specific to a particular CPU or microprocessor family. Low-level languages require a programmer to write instructions for the lowest level of the computer's hardware—that is, for specific hardware elements, such as the processor, registers, and RAM locations. Low-level languages include machine languages and assembly languages.

What is a high-level language? A **high-level language** uses command words and grammar based on human languages to provide what computer scientists call a *level of abstraction* that hides the underlying low-level assembly or machine language. High-level languages, such as COBOL, BASIC, Java, and C, make the programming process easier by replacing unintelligible strings of 1s and 0s or cryptic assembly commands with understandable commands, such as PRINT and WRITE. High-level language commands eliminate many lines of code by substituting a single high-level command for multiple low-level commands (Figure 12-4).

FIGURE 12-4
A single high-level command does the work of multiple low-level commands.

High-level Pascal command

```
Total:=5+4
```

Low-level assembly commands

```
LDA 5
STA Num1
LDA 4
ADD Num1
STA Total
END
```